

SYSTEM AND METHOD FOR IDENTIFYING OPEN  
PERIPHERAL COMPONENT INTERCONNECT (PCI) SLOTS

*Insert Substitute  
Specification TII*

CROSS-REFERENCE TO OTHER APPLICATIONS

*Sub  
a1* This application shares a common specification with U.S.  
Application No. \_\_\_\_\_ filed on January 31, 2001 for a  
5 "System and Method for Assisting in Product Selection."

TECHNICAL FIELD OF THE INVENTION

10 This invention relates generally to computer systems, and  
more particularly to a system and method for identifying open  
peripheral component interconnect (PCI) slots.

BACKGROUND OF THE INVENTION

Computing devices routinely include one or more Peripheral Component Interconnect (PCI) slots. PCI slots are interfaces used by the computer to communicate with peripheral devices, such as Ethernet cards or sound cards, which are inserted or plugged into the PCI slots. The PCI slots typically provide power to the peripheral devices and facilitate communication between the peripheral devices and other components of the computing device.

In order to obtain and install new peripheral devices in the computer, the user often needs to know whether the computing device has any "open" or available PCI slots. For example, a customer who wants to receive Digital Subscriber Line (DSL) service typically needs to obtain a DSL modem, but different DSL modems may communicate with different types of interfaces. The customer needs to determine whether the customer's computer has an available PCI slot. If a PCI slot is available, the customer can order and install a PCI-based DSL modem. Otherwise, the customer needs to order a different type of modem.

Computer users typically have difficulty in identifying whether their computing devices have available PCI slots. For example, identifying available PCI slots typically requires a physical inspection of the computer. Before identifying whether any PCI slots are available, the user often must physically remove at least a portion of the computer casing and locate the PCI slots. It is often a time-consuming process to remove the computer casing, identify the PCI slots, determine if a PCI slot is available, and replace the computer casing. Also, the computer user typically does not possess the technical knowledge needed to identify the available PCI slots. The user may be unaware of what PCI slots look like,

or the user may be unable to distinguish PCI slots from other types of interfaces.

One approach to identifying available PCI slots in a computer is to provide identification software to the user.

- 5 The software helps the user identify the presence or absence of PCI slots by, for example, illustrating the physical steps that the user must take to locate and identify the PCI slots. A problem with this approach is that the user still needs to manually examine the computer, which is a time-consuming process. Also, even with the use of identification software, the user may still incorrectly identify a PCI slot, or the user may still be unable to identify whether a PCI slot is available.
- 10

SUMMARY OF THE INVENTION

The present invention recognizes a need for an improved system and method for identifying open peripheral component interconnect (PCI) slots, which reduce or eliminate some or all of the problems and disadvantages associated with prior systems and methods.

In one embodiment of the invention, a system for identifying an available peripheral component interconnect (PCI) slot in a computing device includes a memory operable to store information identifying at least one PCI slot in the computing device. The system also includes a processor coupled to the memory and operable to identify any PCI devices coupled to a PCI bus. The PCI bus is coupled to the PCI slot. The processor is also operable to determine if any of the identified PCI slots are available without requiring physical inspection of the PCI slots. An available PCI slot comprises an identified PCI slot that is not coupled to an identified PCI device.

In another embodiment of the invention, a method for identifying an available peripheral component interconnect (PCI) slot in a computing device includes identifying at least one PCI slot in the computing device. The method also includes identifying any PCI devices coupled to a PCI bus. The PCI bus is coupled to the PCI slot. The method further includes determining if any of the identified PCI slots are available without requiring physical inspection of the PCI slots. An available PCI slot comprises an identified PCI slot that is not coupled to an identified PCI device.

Numerous technical advantages can be gained through various embodiments of the invention. Various embodiments of the invention may exhibit none, some, or all of the following advantages. For example, in one embodiment of the invention,

a system is provided that simplifies the identification of one or more open PCI slots in a computing device. Rather than requiring a physical inspection of the computing device, the system may detect and report the existence of an open PCI slot to a user. The user may not need to remove the computer's casing, identify the PCI slots, determine if a PCI slot is available, and replace the computer casing. By reducing or eliminating the need for the user to physically inspect the computing device, the system simplifies the identification of open PCI slots and reduces the time it takes to identify whether a PCI slot is available.

Some embodiments of the invention also reduce or eliminate the likelihood that the user will incorrectly identify an open PCI slot. Because the system detects and reports the existence of an open PCI slot, the user may not need to physically inspect the computing device. The user need not know what PCI slots look like or be able to distinguish PCI slots from other types of interfaces. By reducing or eliminating the need for the user to physically inspect the computing device, the system reduces the likelihood that the user will incorrectly identify an open PCI slot.

Other technical advantages will be readily apparent to one of skill in the art from the attached figures, description, and claims.

BRIEF DESCRIPTION OF THE DRAWINGS

For a more complete understanding of the present invention and the advantages thereof, reference is now made to the following descriptions taken in connection with the accompanying drawings, in which:

FIGURE 1 is a block diagram illustrating an exemplary system for identifying open peripheral component interconnect (PCI) slots constructed according to the teachings of the present invention;

FIGURE 2 is a block diagram illustrating an exemplary PCI identification table constructed according to the teachings of the present invention;

FIGURE 3 is a block diagram illustrating an exemplary PCI enumeration table constructed according to the teachings of the present invention;

FIGURE 4 is a flow diagram illustrating an exemplary method for identifying open PCI slots according to the teachings of the present invention;

FIGURE 5 is a flow diagram illustrating one particular example of a method for identifying all PCI slots in a computing device according to the teachings of the present invention;

FIGURE 6 is a flow diagram illustrating one particular example of a method for determining whether identified PCI slots are open according to the teachings of the present invention;

FIGURE 7 is a flow diagram illustrating an exemplary method for identifying a product for use with a computing device according to the teachings of the present invention; and

FIGURE 8 is a block diagram illustrating an exemplary system for identifying a product for use with a computing

device constructed according to the teachings of the present invention.

[illegible]

DETAILED DESCRIPTION OF THE INVENTION

FIGURE 1 is a block diagram illustrating an exemplary system 100 for identifying open peripheral component interconnect (PCI) slots 130. In the illustrated embodiment, system 100 includes a host device 112 having one or more PCI slots 130. In one aspect of the invention, system 100 generates a PCI identification table 138, which identifies all PCI slots 130a-130n in host 112. System 100 also generates a PCI enumeration table 140, which identifies the PCI slots 130a-130n that are coupled to PCI devices. System 100 uses PCI enumeration table 140 to determine which PCI slots 130 listed in PCI identification table 138 are coupled to PCI devices. The PCI slots 130 that are listed in PCI identification table 138 but not in PCI enumeration table 140 are "open" slots, or slots that are available to be coupled to a PCI device in host 112.

Host 112 may execute with any of the well-known MS-DOS, PC-DOS, OS-2, MAC-OS, WINDOWS, UNIX, or other appropriate operating systems. Host 112 may comprise, for example, a desktop computer, a laptop computer, or any other computing or communicating device. In the illustrated embodiment, host 112 includes an input device 114, an output device 116, a hard drive 118, an optical storage media 120 or other appropriate volatile or nonvolatile storage and retrieval devices, a processor (CPU) 122, a read-only memory (ROM) 124, a random access memory (RAM) 126, and one or more controllers 128. Other embodiments of host 112 may be used without departing from the scope of the present invention.

Input device 114 may comprise, for example, a keyboard, mouse, graphics tablet, touch screen, pressure-sensitive pad, joystick, light pen, microphone, or other suitable input device. Output device 116 may comprise, for example, a video



display, a printer, a disk drive, a plotter, a speaker, or other suitable output device.

Processor 122 is coupled to read-only memory 124, random-access memory 126, and controllers 128 by a system bus 142.

5 In this document, the term "couple" refers to any direct or indirect communication between two or more elements, whether or not those elements are in physical contact with one another. Processor 122 is operable to receive information over system bus 142, process the information, and/or  
10 communicate information over system bus 142. Processor 122 may comprise any suitable processor or processors.

Read-only memory 124 is coupled to system bus 142. Read-only memory 124 stores information in system 100, such as data or instructions for processor 122. In one embodiment, read-only memory 124 includes Basic Input/Output System (BIOS)  
15 software and information, such as a PCI Interrupt Request (IRQ) routing table 136. Read-only memory 124 may comprise any of a variety of data structures, arrangements, and/or compilations operable to store and facilitate retrieval of  
20 information. Read-only memory 124 may, for example, comprise a nonvolatile memory.

Random-access memory 126 is coupled to system bus 142. Random-access memory 126 stores information in system 100, such as data used by processor 122. Random-access memory 126  
25 may comprise any of a variety of data structures, arrangements, and/or compilations operable to store and facilitate retrieval of information. Random-access memory 126 may, for example, comprise a volatile memory.

One or more controllers 128 are coupled to system bus  
30 142. Controllers 128 facilitate communication between system bus 142 and one or more peripheral devices coupled to PCI slots 130, a Local Area Network (LAN) interface 132, and/or

Universal Serial Bus (USB) ports 134. In this document, the phrase "peripheral device" refers to any hardware, software, and/or firmware operable to be coupled to a host 112. Peripheral devices include printers, monitors, CD drives, DVD drives, mice, keyboards, sound cards, modems, and any other suitable devices. Controllers 128 may, for example, receive information over a PCI bus 144 from a peripheral device coupled to a PCI slot 130. Controllers 128 may also receive information over system bus 142 and communicate the information over a USB bus 146 to a peripheral device coupled to a USB port 134. One controller 128 may be used to communicate over PCI bus 144, USB bus 146, and/or other busses, or separate controllers 128 may be used. Controllers 128 may comprise any hardware, software, firmware, or combination thereof operable to facilitate communication between host 112 and a peripheral device. In one embodiment, controllers 128 include a PCI controller for communicating over PCI bus 144 and a USB controller for communicating over USB bus 146. Controllers 128 could also include any other suitable controller, such as a Small Computer System Interface (SCSI) controller.

PCI slots 130 are coupled to PCI bus 144. Each PCI slot 130 is operable to be coupled to a PCI device. In this document, the phrase "PCI device" refers to any hardware, software, and/or firmware operable to communicate over PCI bus 144. PCI devices include printers, monitors, CD drives, DVD drives, mice, keyboards, sound cards, modems, hard drives, or any other suitable devices. PCI slot 130 also facilitates communication between the PCI device and PCI bus 144. PCI slot 130 may comprise any hardware, software, firmware, or combination thereof operable to facilitate communication between a PCI device and PCI bus 144.

LAN interface 132 is coupled to controllers 128. LAN interface 132 facilitates communication between controllers 128 and a network. LAN interface 132 may comprise any hardware, software, firmware, or combination thereof operable to facilitate communication between host 112 and a network. LAN interface 132 may, for example, comprise an Ethernet card.

USB ports 134 are coupled to USB bus 146. USB port 134 is operable to be coupled to one or more peripheral devices and to facilitate communication between the peripheral devices and USB bus 146. USB port 134 may comprise any hardware, software, firmware, or combination thereof operable to facilitate communication between a peripheral device and USB bus 146.

System bus 142 is coupled to processor 122, read-only memory 124, random access memory 126, and controllers 128. System bus 142 facilitates communication between processor 122, read-only memory 124, random access memory 126, and controllers 128. System bus 142 may, for example, transport data and instructions between processor 122 and controllers 128. System bus 142 may comprise any suitable bus operable to transfer information between elements in host 112.

PCI bus 144 is coupled to controllers 128 and one or more PCI slots 130. PCI bus 144 is a local bus that facilitates communication between controllers 128 and peripheral PCI devices coupled to PCI slots 130. PCI bus 144 may comprise any suitable local bus operable to transfer information between controllers 128 and PCI slots 130.

USB bus 146 is coupled to controllers 128 and one or more USB ports 134. USB bus 146 is a bus that facilitates communication between controllers 128 and external peripheral devices through USB ports 134. USB bus 146 may comprise any

suitable bus operable to transfer information between controllers 128 and USB ports 134.

In one aspect of operation, read-only memory 124 includes PCI Interrupt Request (IRQ) routing table 136. PCI IRQ routing table 136 includes information that identifies each PCI slot 130 in host 112. PCI IRQ routing table 136 may, for example, include information identifying PCI bus 144, a device number for each PCI slot 130 coupled to PCI bus 144, and a slot number for each PCI slot 130 coupled to PCI bus 144. In one embodiment, PCI IRQ routing table 136 is identified by a "\$PIR" string at the beginning of the table. In that particular example, PCI IRQ routing table 136 is stored in read-only memory 124 between memory addresses F0000h and FFFFFh (hexadecimal). Other identifying strings and/or memory locations may be used without departing from the scope of the present invention.

To determine whether any PCI slots 130 are available in host 112, processor 122 generates a PCI identification table 138 and a PCI enumeration table 140. To generate PCI identification table 138, processor 122 locates PCI IRQ routing table 136 in read-only memory 124. Processor 122 may, for example, scan read-only memory 124 for a "\$PIR" string, which represents the beginning of PCI IRQ routing table 136. Processor 122 accesses the information in PCI IRQ routing table 136, extracts the information about PCI slots 130, and generates PCI identification table 138. The information stored in PCI identification table 138 identifies PCI slots 130a-130n in host 112. To generate PCI enumeration table 140, controllers 128 enumerate PCI devices coupled to PCI bus 144, which includes identifying PCI slots 130 that are coupled to PCI devices. In one embodiment, host 112 executes with a WINDOWS NT operating system, and system 100 enumerates PCI

devices coupled to PCI bus 144 using a HalGetBusData function. In another embodiment, host 112 executes with a WINDOWS 98 operating system, and system 100 enumerates PCI devices using a system registry. Other methods of enumerating devices on PCI bus 144 may be used without departing from the scope of the present invention. After enumerating devices coupled to PCI bus 144, processor 122 compares PCI identification table 138 and PCI enumeration table 140. Processor 122 identifies which PCI slots 130 are not coupled to PCI devices, and these PCI slots 130 are "open" or available to be coupled to a PCI device.

The ability to identify open PCI slots 130 may be useful in many ways, such as in determining what product to install in computer 112. In this document, the term "product" refers to any hardware, software, and/or firmware that may be coupled to or used in conjunction with host 112. For example, a modem could be coupled to a PCI slot 130, LAN interface 132, or a USB port 134. In one embodiment, processor 122 may identify whether host 112 includes any available PCI slots 130. Processor 122 may also determine whether host 112 includes a LAN interface 132 and/or a USB port 134. Using this information, processor 122 may recommend the type of modem that may be installed in host 112. In one embodiment, processor 122 may recommend a USB modem if host 112 supports the USB standard. If not, processor 122 may determine whether any PCI slots 130 are open and recommend a PCI-based modem if a PCI slot 130 is available. Otherwise, processor 122 may determine whether a LAN interface 132 is present and recommend a LAN-based modem. If processor 122 cannot identify a suitable type of product, processor 122 may instruct the user to contact a technician or customer support personnel.

Although FIGURE 1 illustrates one embodiment of system 100, various changes may be made without departing from the scope of the present invention. For example, LAN interface 132 could be coupled to USB bus 146 instead of PCI bus 144. Also, any number of PCI slots 130, LAN interfaces 132, and/or USB ports 134 may be used in host 112. Further, PCI IRQ routing table 136, PCI identification table 138, and/or PCI enumeration table 140 may reside in any memory internal to or accessible by host 112. In addition, although system 100 illustrates host 112 as comprising a desktop computer, host 112 may comprise any suitable computing device, such as a laptop computer or server. Other changes may be made without departing from the scope of the present invention.

FIGURE 2 is a block diagram illustrating an exemplary PCI identification table 238. In this document, the term "table" refers to any arrangement, compilation, and/or data structure operable to store and facilitate retrieval of information. PCI identification table 238 may be useful, for example, in system 100 of FIGURE 1. In the illustrated embodiment, PCI identification table 238 includes at least one entry 250. An entry 250 in PCI identification table 238 corresponds to a PCI slot 130 in system 100, and each entry 250 includes a bus number 252, a device number 254, and a slot number 256. The values and number of entries 250 contained in PCI identification table 238 are for illustration only. Any suitable number of entries 250 and any suitable values may be used without departing from the scope of the present invention.

Bus number 252 identifies PCI bus 144 in system 100. In the illustrated embodiment, PCI bus 144 is identified by the number zero, although PCI bus 144 may be identified using any suitable indicator. Also, if multiple PCI busses 144 are used

in system 100, each bus 144 may be identified by a different bus number 250.

Device number 254 identifies a device number assigned to each PCI slot 130 in host 112. A device number 254 may be used, for example, to address a device coupled to PCI slot 130. In one embodiment, device numbers 254 are assigned to PCI slots 130 by the BIOS of host 112. PCI slots 130 may be assigned any suitable device number 254. In one embodiment, different device numbers 254 may be used to identify the same PCI slot 130. For example, when a single hardware device performs multiple functions, such as a sound card having a SCSI controller, two device numbers 254 could be assigned to the PCI slot 130 that is coupled to the sound card.

Slot number 256 identifies whether an entry 250 in PCI identification table 238 identifies a system-board device or a PCI slot 130. A slot number 256 of zero indicates that entry 250 corresponds to a system-board device, or a device that is hardwired to the motherboard of host 112. A slot number 256 having a non-zero value represents an "add-in slot," or a PCI slot 130 in host 112 that can be coupled to a PCI device. In one embodiment, an entry 250 having a slot number 256 of zero is not an available PCI slot 130, and an entry 250 having a non-zero slot number 256 may or may not be an available PCI slot 130.

In one embodiment, processor 122 generates PCI identification table 238 using information from PCI IRQ routing table 136. For example, processor 122 may scan read-only memory 124 for a "\$PIR" string, which represents the beginning of PCI IRQ routing table 136. After locating table 136 in read-only memory 124, processor 122 may extract the bus number 252, device number 254, and slot number 256 for each

entry in PCI IRQ routing table 136. Using this information, processor 122 may generate PCI identification table 238.

Although FIGURE 2 illustrates one embodiment of PCI identification table 238, other embodiments may be used without departing from the scope of the present invention. For example, although FIGURE 2 illustrates a table 238 storing the information in entries 250, other data structures may be used in system 100. The information from entries 250 could, for example, be stored in a linked list, multiple arrays, or any other suitable data structure operable to store and facilitate retrieval of information.

FIGURE 3 is a block diagram illustrating an exemplary PCI enumeration table 340. PCI enumeration table 340 may be useful, for example, in system 100 of FIGURE 1. In the illustrated embodiment, PCI enumeration table 340 includes zero or more entries 350. An entry 350 in PCI enumeration table 340 corresponds to a PCI device coupled to PCI bus 144 in system 100. Each entry 350 in PCI enumeration table 340 includes a bus number 352 and a device number 354. The values and number of entries 350 contained in PCI enumeration table 340 are for illustration only. Any suitable number of entries 350 and any suitable values may be used without departing from the scope of the present invention.

Bus number 352 identifies PCI bus 144 in system 100. Device number 354 identifies a device number assigned to a PCI device coupled to PCI bus 144. Device numbers 354 in entries 350 may have any suitable value, and different device numbers 354 may be used to identify the same PCI device.

In one embodiment, processor 122 generates PCI enumeration table 340 using information collected by controllers 128. For example, controllers 128 may enumerate PCI devices coupled to PCI bus 144 using a HalGetBusData



function or a system registry. During enumeration, controllers 128 identify the PCI devices coupled to PCI bus 144. For each device, controllers 128 identify the bus number 352 of the device and the device number 354 of the device.  
5 Processor 122 collects this information and generates PCI enumeration table 340.

Although FIGURE 3 illustrates a table 340 storing the information in entries 350, other data structures may be used to store the information without departing from the scope of the present invention. The information from entries 350 could, for example, be stored in a linked list, multiple arrays, or any other suitable data structure operable to store and facilitate retrieval of information. Also, because the number of entries 350 in PCI enumeration table 340 is related to the number of PCI devices coupled to PCI bus 144, table 340 may contain any number of entries 350.  
10  
15

In one aspect of operation, processor 122 generates PCI identification table 238 and PCI enumeration table 340. To identify open PCI slots 130 in host 112, processor 122 may compare entries 250 and 350 in tables 238 and 340. In one embodiment, processor 122 compares the bus number 352 and device number 354 of an entry 350 in PCI enumeration table 340 with the entries 250 in PCI identification table. If an entry 250 in table 238 has a bus number 252 and a device number 254 that matches the bus number 352 and device number 354 of entry 350 in table 340, entry 250 is removed from table 238. In this instance, entry 250 does not correspond to an open PCI slot 130 because an enumerated PCI device is present in the PCI slot 130. Processor 122 repeats this process for each entry 350 in table 340, and any remaining entries 250 in table 238 represent open PCI slots 130 in system 100. Alternatively, instead of removing entries 250 from table 238,  
20  
25  
30

system 100 could construct a new table that stores information identifying open PCI slots 130.

Using the tables of FIGURES 2 and 3 as an example, entry 350k has a bus number 352 of "0" and a device number 354 of "f". Processor 122 compares bus number 352 and device number 354 to entries 250 in table 238 of FIGURE 2, and processor 122 finds that entry 250l in table 238 has a bus number 252 of "0" and a device number 254 of "f". Entry 250l is removed from table 238 because it is not an available PCI slot 130. After repeating this process for each entry 350, only entry 250k would remain in table 238. Entry 250k corresponds to slot 3, which is an available PCI slot 130. Because slot 3 is present in host 112 but is not coupled to a PCI device, slot 3 is an open PCI slot 130.

In the illustrated embodiment, entry 250n has a slot number 256 of zero. An entry 250 having a slot number 256 of zero corresponds to a hardwired device on the motherboard of host 112. That entry 250 does not represent an available PCI slot 130. In one embodiment, processor 122 could allow entry 250n to remain in table 238. Entry 250n would eventually be removed from table 238 when processor 122 compares it to entry 350m. In another embodiment, system 100 could remove entries 250 having a slot number 256 of zero before comparing entries 250 and 350.

In the illustrated embodiment, entry 350a in table 340 contains values of zero for bus number 352 and device number 354. In one embodiment, devices such as a PCI bus controller 128 may be enumerated as a device on PCI bus 144. In a particular embodiment, controller 128 may be enumerated as having a bus number 352 of zero and a device number 354 of zero. The controller 128 is not a physical PCI slot 130 in

host 112, so system 100 may ignore this entry 350 in table 340 when identifying open PCI slots 130 in host 112.

FIGURE 4 is a block diagram illustrating an exemplary method 400 for identifying open PCI slots 130. System 100 identifies PCI slots 130 in host 112 at step 500. This may include, for example, processor 122 scanning read-only memory 124 and locating PCI IRQ routing table 136. This may also include processor 122 extracting a bus number 252 and a device number 254 for each PCI slot 130 from PCI IRQ routing table 136, and storing the bus numbers 252 and device numbers 254 in PCI identification table 238. System 100 identifies PCI devices coupled to PCI bus 144 at step 550. This may include, for example, controllers 128 enumerating the PCI devices by identifying a bus number 352 and a device number 354 of any PCI device coupled to PCI bus 144. This may also include processor 122 storing the bus numbers 352 and device numbers 354 in PCI enumeration table 340.

System 100 determines, without requiring physical inspection of the PCI slots 130, which identified PCI slots 130 are available at step 600. This may include, for example, processor 122 comparing the entries 250 in PCI identification table 238 and the entries 350 in PCI enumeration table 340. This may also include processor 122 identifying any entry 250 in PCI identification table 238 that does not have a matching entry 350 in PCI enumeration table 340.

Although FIGURE 4 illustrates one embodiment of a method 400 for identifying open PCI slots 130, various changes may be made to method 400 without departing from the scope of the present invention. For example, system 100 may identify PCI devices coupled to PCI bus 144 before identifying PCI slots 130 in host 112.

FIGURE 5 is a flow diagram illustrating one particular example of a method 500 for identifying all PCI slots 130 in a computing device 112. System 100 scans a memory for a PCI IRQ routing table 136 at step 502. This may include, for example, processor 122 scanning read-only memory 124 between memory addresses F0000h and FFFFFh (hexadecimal). System 100 locates the PCI IRQ routing table 136 at step 504. This may include, for example, processor 122 locating a "\$PIR" string in read-only memory 124.

System 100 determines the number of entries in PCI IRQ routing table 136 at step 506. In one embodiment, PCI IRQ routing table 136 includes an entry that defines the size of PCI IRQ routing table 136 in bytes. Also, PCI IRQ routing table 136 may contain thirty-two bytes of information in addition to the entries, and each entry may contain sixteen bytes of information. In that particular example, processor 122 may determine the number of entries in PCI IRQ routing table 136 using the formula:

$$\text{Number of entries} = (\text{Size of table} - 32) / 16.$$

The number of entries represents the number of PCI slots 130 and system-board devices coupled to PCI bus 144 in host 112.

System 100 identifies the bus number 252 and device number 254 for each entry at step 508. This may include, for example, processor 122 scanning PCI IRQ routing table 136 and extracting the bus number 252 and device number 254 for each entry. System 100 may store this information in PCI identification table 238 or any other suitable data structure.

Although FIGURE 5 illustrates one embodiment of a method 500 for identifying PCI slots 130 in a host 112, various changes may be made to method 500 without departing from the

scope of the present invention. For example, system 100 may identify the number of entries in PCI IRQ routing table 136 during or after the scan of table 136, such as by counting the number of entries found in table 236. Also, system 100 may  
5 extract additional information about each PCI slot 130 from PCI IRQ routing table 136, such as a slot number 256 for each entry.

FIGURE 6 is a flow diagram illustrating one particular example of a method 600 for determining whether identified PCI  
10 slots 130 are open. System 100 initializes an open slot counter at step 602. This may include, for example, processor 122 setting the counter to a value equal to the number of entries 250 in table 238. System 100 removes any identified PCI slots 130 having a slot number of zero at step 604. This  
15 may include, for example, processor 122 scanning table 238 and removing any entries 250 having a slot number 256 of zero since these entries 250 do not correspond to an open PCI slot 130. This may also include processor 122 decrementing the open slot counter for each entry 250 removed. System 100  
20 selects the first enumerated PCI device at step 606. This may include, for example, processor 122 selecting the first entry 350 in PCI enumeration table 340.

System 100 compares the enumerated PCI device to the identified PCI slots 130 at step 608. This may include, for  
25 example, processor 122 comparing the bus number 352 and device number 354 of entry 350 to the bus numbers 252 and device numbers 254 of entries 250. If system 100 finds an entry 250 having a bus number 252 and device number 254 that matches the bus number 352 and device number 354 of entry 350, system 100  
30 identifies a match at step 609. In that case, the PCI slot 130 corresponding to entry 250 is not an available slot because the PCI slot 130 is coupled to a PCI device. System

100 removes the identified PCI slot 130 at step 610. This may include, for example, processor 122 removing entry 250 from table 238. System 100 also decrements the open slot counter at step 612.

5 After decrementing the open slot counter at step 612, or if system 100 found no match at step 609, system 100 determines whether another enumerated PCI device exists at step 614. This may include, for example, processor 122 determining whether another entry 350 exists in PCI enumeration table 340. If another PCI device exists, system 100 selects the next PCI device at step 616 and returns to step 608 to process this PCI device.

10 If no more enumerated PCI devices exist at step 614, method 600 ends. At this point, any remaining entries 250 in table 238 identify open or available PCI slots 130. Also, the value of the open slot counter should equal the number of available PCI slots 130.

15 Although FIGURE 6 illustrates one embodiment of a method 600 for determining whether identified PCI slots 130 are open, various changes may be made without departing from the scope of the present invention. For example, entries 250 and 350 in tables 238 and 340 could be sorted in order of increasing or decreasing device number 254 and 354 to simplify the determination at step 609. Also, in another embodiment, step 20 604 could be omitted because devices having a slot number 256 of zero should appear in PCI enumeration table 340 as enumerated PCI devices. Further, in the illustrated embodiment, system 100 compares the bus number 352 and device number 354 of each enumerated PCI device to the bus numbers 25 252 and device numbers 254 of identified PCI slots 130. In another embodiment, system 100 may compare the bus number 252 and device number 254 of each identified PCI slot 130 to the

bus numbers 352 and device numbers 354 of enumerated PCI devices.

FIGURE 7 is a flow diagram illustrating an exemplary method 700 for identifying a product for use with a computing device 112. The method 700 begins at step 702 where system 100 identifies the operating system of host 112. This may include, for example, processor 122 determining whether host 112 is executing with WINDOWS 98 or WINDOWS 98 Second Edition, or whether host 112 is executing with another operating system. System 100 determines whether the operating system of host 112 supports the use of a USB bus 146 at step 704. For example, the WINDOWS 98 operating system supports a USB bus 146, while the WINDOWS 95, WINDOWS NT 3.51, and WINDOWS NT 4.0 operating systems do not.

If the operating system supports the use of a USB bus 146, system 100 determines if a USB root hub is present at step 706. This may include, for example, processor 122 accessing a system registry to determine if a USB root hub is installed on the motherboard of host 112 or attached to host 112 as an adapter card. If a USB root hub is present, system 100 identifies a USB-based product as available for recommendation at step 708. Alternatively, system 100 could automatically recommend a USB-based product at this point. A USB device may be coupled to a USB port 134 and used with host 112.

System 100 can also determine whether any PCI slots 130 are available at step 710. This may include any method of determining whether a PCI slot 130 is open, such as processor 122 executing method 400 illustrated in FIGURE 4. If any PCI slots 130 are available, system 100 identifies a PCI-based product as available for recommendation at step 712. Alternatively, system 100 could automatically recommend a PCI-

based product at this point. A PCI device may be coupled to a PCI slot 130 and used with host 112.

System 100 can further determine if a LAN interface 132 is available for use in host 112 at step 714. This may include, for example, system 100 prompting a user to enter whether host 112 includes a LAN interface 132, or processor 122 automatically detecting the presence or absence of LAN interface 132. If a LAN interface 132 is present, system 100 identifies a LAN-based product as available for recommendation at step 716. Alternatively, system 100 could automatically recommend a LAN-based product at this point. If no product has been identified, method 700 ends without recommending a product. A user of host 112 may be advised to contact a technician or customer support personnel for assistance in identifying the type of product to install in host 112.

Although FIGURE 7 illustrates one embodiment of a method 700 for identifying a product for use with a computing device 112, various changes may be made to method 700 without departing from the scope of the present invention. For example, system 100 may perform testing steps 704, 706, 710, and 714 in any order. In another embodiment, system 100 may determine whether a LAN interface 132 is available before testing whether any PCI slots 130 are available. Also, while FIGURE 7 illustrates system 100 testing for three types of interfaces, system 100 may check whether any number of interfaces are available. For example, system 100 may determine whether any USB ports 134 or PCI slots 130 are available, without determining if a LAN interface 132 is present. In addition, system 100 may take any suitable action after identifying one or more available interfaces. In one embodiment, system 100 may present a user with a list of all identified products and allow the user to select one or more



of the products. In another embodiment, system 100 may recommend one or more of the identified products to the user using any suitable ranking or rating criteria. Other changes may be made to method 700 without departing from the scope of the present invention.

The process described in FIGURE 7 could be hardwired into host 112 or could exist all or in part as an application 148 residing on host 112 or otherwise accessible to host 112. Where all or a part of the process is executed through application 148, host 112 could access application 148, for example, by loading application 148 from a CD, a floppy disk, or any other computer readable medium. Alternatively, host 112 could access an application 148 residing at a remote site, such as a network server.

FIGURE 8 is a block diagram illustrating an exemplary system 800 for identifying a product for use with a computing device 812. In the illustrated embodiment, system 800 includes a host 812 and a server 850 coupled to a network 852. Other embodiments of system 800 may be used without departing from the scope of the present invention.

Host 812 and server 850 are coupled to network 852. Host 812 may be the same or similar to host 112 of FIGURE 1. Server 850 is operable to communicate with host 812 and to identify a product for use with host 812. In one embodiment, an application 848 resides at server 850, and host 812 downloads and executes the application 848 to identify available interfaces in host 812 and/or identify a product for use with host 812. In one embodiment, server 850 includes a web portal 854, and a user of host 812 may access the web portal 854 through network 852. Server 850 may comprise any suitable hardware, software, firmware, or combination thereof operable to communicate with host 812 over network 852.

Network 852 is coupled to host 812 and server 850 by communication links 856. Network 852 may include any suitable wireline or wireless system that supports communication between network elements using ground-based and/or space-based components. For example, network 852 may be a public switched telephone network (PSTN), an integrated services digital network (ISDN), a local area networks (LAN), a wide area network (WAN), a global computer network such as the Internet, or any other communications system or systems at one or more locations.

Communication links 856 facilitate communication between host 812 or server 850 and network 852. Communication link 856 may comprise any communications medium operable to facilitate communication of analog and/or digital signals using ground-based and/or space-based components. Communication link 856 may, for example, comprise a twisted-pair copper telephone line or a fiber optic line.

In one embodiment, a user of host 812 may access web portal 854 of a product supplier and download an application 848 to host 812. The application 848 may be the same or similar to application 148 of FIGURE 1. The user may choose to execute the application 848, or the application 848 may be a self-executing application. The application 848 may then identify the types of interfaces available in host 812. In a particular embodiment, application 848 identifies whether any PCI slots, LAN interfaces, and/or USB ports are available in host 812. Application 848 uses this information to identify or recommend one or more products for use with host 812. The application 848 may, for example, use the same or similar method illustrated in FIGURE 7 for identifying a product.

Application 848 may take any appropriate steps after identifying the types of interfaces available in host 812.

The application 848 may, for example, present the user with a list of all identified products and allow the user to select one or more of the products. Application 848 may also recommend one of the identified products to the user, or application 848 may select a product without user input. In addition, after a product is selected, application 848 may communicate an identification of the selected product to server 850. Server 850 may then take any suitable action, such as generating an order for the product, collecting billing information from the user, estimating a delivery time for the product, and/or informing the user of the estimated delivery time.

As a particular example, host 812 may include a 56K modem 858, and the user of host 812 may wish to obtain a Digital Subscriber Line (DSL) modem from a DSL service provider. The user may access web portal 854 of the DSL service provider and download application 848 to host 812. Host 812 may execute the application 848, and the application 848 identifies the types of interfaces available in host 812. The application 848 may also identify one or more types of DSL modems for use with host 812, recommend a type of DSL modem, or automatically select a modem. Application 848 may also communicate the selected modem type to the web portal 854, and delivery of the DSL modem can be arranged. In one embodiment, application 848 may treat the interface coupled to modem 858 as an available slot and instruct the user of host 812 to insert the DSL modem into the slot occupied by the 56K modem.

Although FIGURE 8 illustrates one embodiment of a system 800 for identifying a product for use with a computing device 812, various changes may be made to system 800 without departing from the scope of the present invention. For example, although application 848 is described as identifying

and/or recommending a product for use with host 812, this functionality may be divided between application 848 and server 850. In one embodiment, application 848 may identify the types of interfaces available in host 812 and communicate this information to server 850. In this embodiment, server 850 may identify one or more products for use with host 812 and communicate this information to host 812. Also, application 848 may be delivered to host 812 using any suitable method. As an example, application 848 may be placed on a CD, which is then placed into a CD drive in host 812. Other changes may be made to system 800 without departing from the scope of the present invention.

Although the present invention has been described in several embodiments, a myriad of changes, variations, alterations, transformations, and modifications may be suggested to one skilled in the art, and it is intended that the present invention encompass such changes, variations, alterations, transformations, and modifications as fall within the spirit and scope of the appended claims.